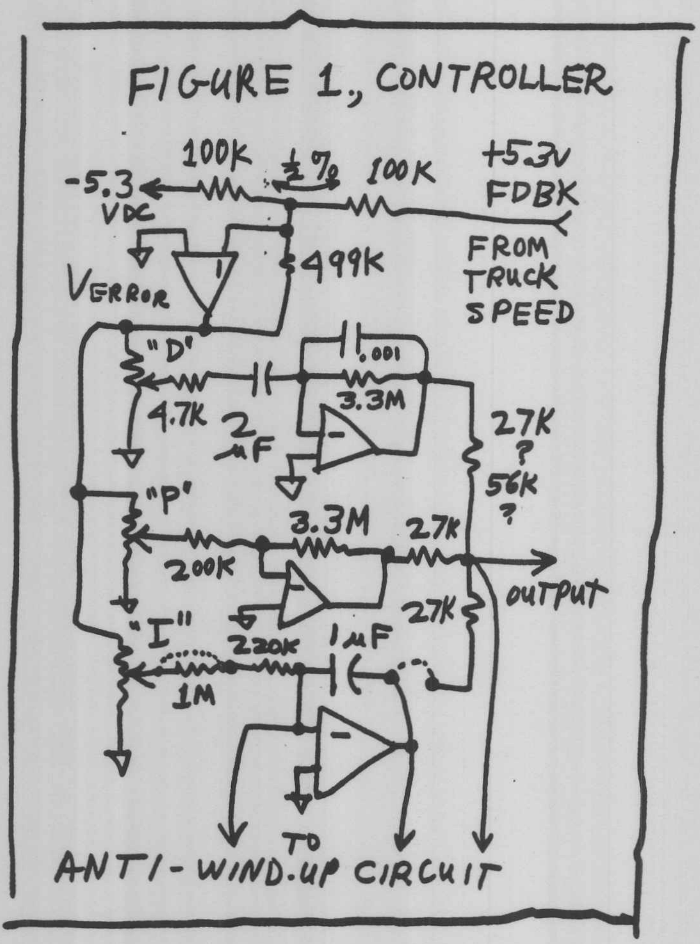
BOB PEASE *obtained a BSEE from MIT in 1961 and is Staff Scientist at National Semiconductor Corp., Santa Clara, Calif.*

What's All This Fuzzy Logic Stuff, Anyhow? (Part IV)

Recently, one of my far-flung friends sent me a magazine article, along with his puzzled comment: "Can Fuzzy Logic REALLY do THIS—and nobody else can do it?" I read the story. It explained that in Europe, there was a requirement for a speed limiter for heavy trucks, at 53.4 mph (86 km/hr). The article claimed that such a speed limiter couldn't be done by any *conventional controller*.

This was partly because it was very hard to design a model for a truck. There were so many different kinds and versions and manufacturers of trucks. Plus, there were heavy trucks and lightly loaded modes of operation. The truck model would have to work on upgrades and downgrades. Also, the task was claimed to be very nonlinear. So, the author proposed using Fuzzy Logic (FL) to accomplish this. Because his controller would be very versatile and very robust, it wouldn't need to be programmed differently for different types of trucks.



Now, if this article said there were something that I couldn't do with op amps, resistors, and capacitors, then that would present kind of a serious challenge, eh? I always like challenges.

I thought about this. A truck tends to accelerate over a wide range of speeds. It picks up speed, faster or slower, according to its load, proportional to its accelerator-pedal setting, and based on how the engine's torque is geared to the load and influenced by the grade of the road. That sounds like an *integrator*—an integrator with a larger or smaller feedback capacitor (to represent how the mass of the truck changes), with a positive bias (in case it's "on a downgrade"), or with a negative bias (in case it's "on an upgrade").

A truck's *model* sounds pretty easy to me—an integrator with some biases, varying gains, and a little lag in the control path. I could then use the model to drive an analog controller like the one shown in Figure 1. Now, I concede that it would be hard to model a truck for every situation, at all speeds. But in this case, a model that's fairly realistic at around 45 to 55 mph is easy to design. See Figure 2—an integrator with a controller input (with a little lag). What's so difficult about that?

So I sat down with a big sheet of paper, added a few control elements to a simple PID loop, proposed some nominal component values, and drew up a couple of wires to connect this to the model of the truck *(Fig. 1, again)*. This is very much like the controller for my "Ball-On-Beam Balancer" (ELECTRONIC DESIGN ANALOG SUPPLEMENT, *Nov. 20, 1995, p. 50)*.

Next, I started to draw up some waveforms to show how it would work. In my book,[1] I call this a "choreography" to illustrate how each waveform relates to one another. I convinced myself that this loop could be stable and wouldn't oscillate. This is largely related to the way that a control loop can be fairly SIMPLE, NIMBLE, and quick to *increase* or *decrease* gas-pedal settings. The truck can accelerate at about 0.01 to 0.05 G's if it's heavily loaded, or perhaps at five or 10 times that rate if it's empty. ANY driver can easily pull his foot off the gas pedal when he wants to avoid exceeding a speed limit—*if* he wants to, and *if* he remembers to. Even an FL controller can do that. Even a PID controller can do that. Controlling the speed of a vehicle isn't too difficult. Anybody can do it.

That's because a truck trying to accelerate with a nominal amount of power, driving a wide-range load, is only an integrator, just like an op amp. Every-

it's easy to close a loop around them—whether at high gain, or at low gain—because the phase-shift of an integrator is 90°. Therefore, the loop will be stable. But then I saw the flaw in my first-draft controller circuit: The PID controller has probably been trying to accelerate at full power for a long time, because the vehicle's speed is below the limit. The integrator has, thus, been fed a large dc error signal for a long time—and its output would be near the negative limit! That could easily cause overshoot, and the loop couldn't help it. Overshoot would occur, even though stable operation would eventually take over.

How can I avoid this overshoot? After all, in a speed limiter, overshoot is a really bad kind of performance. The requirement for this limiter is pretty strict, allowing only a small amount of overshoot (about 3 mph max).

Then I remembered some notes by Dave St. Clair[2] about "wind-up." Any PID controller can have its integrator term go off to a limit when you don't want it to, such as when the loop isn't in control for a long time. That's called wind-up. How do we avoid this? After all, *any* PID controller tends to exhibit this wind-up, if it's tasked to pull a large inertial MASS up to a new level. This is NOT a new or unique task. Actually, some PID controllers do very well at this!

I recalled that the remedy is termed "anti-wind-up." I tried looking this up in several books. I found it really hard to find any further mention of wind-up, or of how to perform anti-wind-up. Still, I figured out that it isn't rocket science! This is a special case of analog computation—with a bit of hybrid/digital control.

Then I remembered seeing some sketches that had the same shape *(Fig. 3)* as my crude first-hack PID controller with wind-up. In the old book by MacNeill and Freiberger, *Fuzzy Logic*,[3] about the origins of FL in control loops, there was that sketch on page 115 showing the performance of a little boiler. McNeill said that the "conventional controller" would cause overshoot, whereas an FL controller, designed by Mr. Mamdani, seemed to avoid overshoot. Ahem! I realized that this was an example where a PID controller was said to give poor performance—because it was MIS-applied. It had a big overshoot, a result of its integrator term having a lot of *wind-up*. If I were foolish enough to disregard wind-up, then my controller would show bad overshoot too!

Why didn't Mamdani's FL controller have a lot of wind-up? Aha—it didn't have any integrator term. Then why did it slow down and converge on the right answer without overshoot? It was easy to see that was due to the FL controller cutting its gain back severely, as it approached the null—because it ran out of gain.

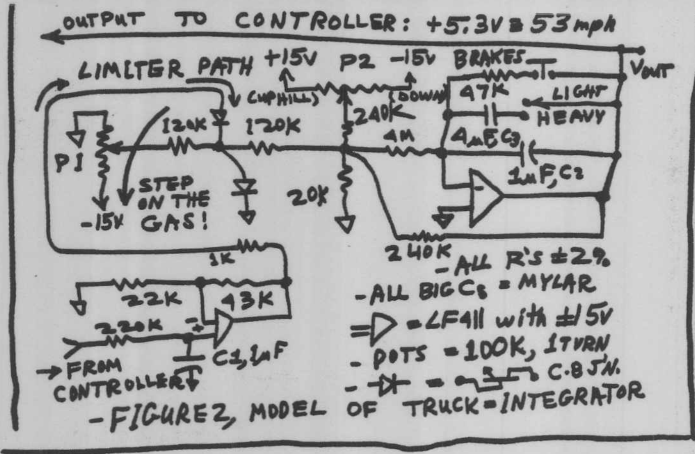
The example showed that the FL con- and slow as it approached the target. But the book didn't show how that controller would react when a load was placed on the boiler to draw off steam, so that it could do some useful WORK. I could see what was going on: the FL system didn't have much gain, and it would look good only if you didn't put on a load. The FL controller had P and D terms, but it didn't have any integrator to provide high gain against a dc load.

The FL engineers praised one good feature of their controller—the start-up—in comparison to a MIS-applied conventional (PID) controller. Yet, they avoided mentioning the situation where their FL controller would show its deficiency. It would bog down and allow the boiler pressure to drop by 10% or more when the boiler had to do some *work*.

Any good PID controller would NOT let the boiler pressure drop under a load. Instead, after a brief transient drop of pressure, the PID would pull the pressure right back to the ideal pressure, with no dc errors. If properly applied with anti-wind-up, therefore, the PID controller could easily perform well at startup, without overshoot, and with good regulation against loads. Have you ever seen any FL controllers that were able to do that?

Now, after learning about this problem, could I design a controller for a truck's speed limiter? Sure. Could I invent a decent anti-wind-up circuit? Certainly *(Fig. 4)*. The detector can tell if the controller's error is far from null, and it applies a RESET switch across the integrator. The integrator is reset to zero output, until the controller begins to take effect, and then the integrator in the controller begins to servo out the dc error. Of course, this circuit is much more complicated and more versatile than I would really require. But, I wanted to conduct some experiments.

Do I have confidence that this controller will work



-FIGURE 2, MODEL OF TRUCK=INTEGRATOR

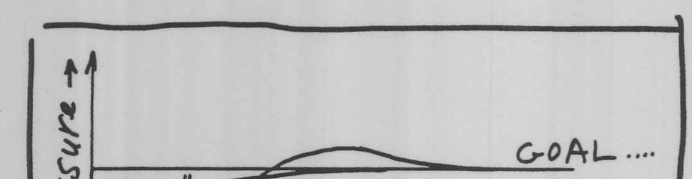


Figure 3. Results of Steam Pressure Controllers ....

well, and won't overshoot? I definitely do. I don't know how the FL controller was designed, and I don't know what features it has. I don't know how much it would slow down under load either. But I'm convinced that my speed limiter would work well, would NOT overshoot, hunt, ring, or oscillate, and would work well under all conditions of light, medium, or heavy loads, as well as on upgrades, downgrades, and flat terrain. Plus, it wouldn't work so jerkily as to make the truck driver grouchy. It would have high accuracy, with high gain and quick response. It really would be interesting to compare it to an FL controller. But I should be able to show you the controller's performance in the next issue of *Electronic Design*.

One guy asked me, "Bob, why do you call this a controller, when the requirement is for a speed *limiter*?" I thought about it and explained, "This circuit can do EITHER. To turn it into a controller—just like a cruise control in a car—simply *reverse* the diode that goes to the two 120-kΩ resistors in Figure 2."

Furthermore, it would be easy to make it work on just about any truck. The good news is that most trucks, if they are heavily loaded so as to have markedly different dynamics, are SLOWER. It might be hard to design a speed controller for a very fast vehicle, under all of those conditions. It may be difficult if a four-ton truck were changed to a weight of one ton, instead of loaded to 20 or 40 tons. But if my controller will work on a lightly loaded truck, it will EASILY work on a heavily loaded truck, with slow acceleration. The wide range doesn't make it hard to control the loop—it makes it *easier*.

Of course, if the truck is on a downgrade, the controller will have to pull the throttle nearly closed. On an upgrade, the controller will have to allow the throttle to stay nearly wide-open without much error. Still, that's not a big deal for a decent PID controller.

Have I built this controller? Not yet. But I bought some parts, some one-turn pots and 1-μF capacitors. I grabbed a bunch of LF412s to make it simple to cover a wide dynamic range, with ±12-V signals. When I'm finished, this function will probably require only one or two dual op amps to make the controller, in the ultimate simplified version, all running on +5 V dc.

Will this controller work for all trucks? Yeah, I think so. It's difficult to imagine a truck that wouldn't be controllable by this controller. (If there were such a truck, it couldn't be controlled by a *driver*.)

More comments will come later. We'll let you know how this all works out in two weeks. Meanwhile, I must say that I often agree with the author—that he usually shows us some very good work with FL. But that's not always the case—not here.

So, we finally figured out why the FL guys keep stating that conventional controllers are really hard to apply. We realized that they say the *conventional controller* would work badly because the system was claimed to be very nonlinear. The reason behind this was a misunderstanding of PID. Also, they only made a comparison to PID controllers that are MIS-applied, such as without anti-wind-up, as I suspected all along.

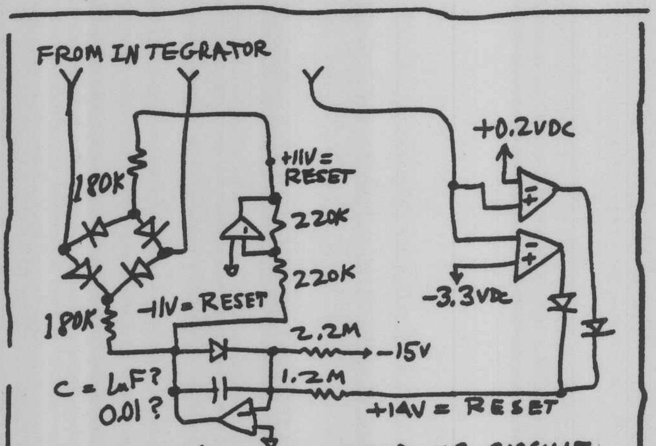


FIGURE 4, ANTI-WIND-UP CIRCUIT

I have been assured that *Electronic Design* will give any promoters of Fuzzy Logic a chance to respond to my columns in print. But save THIS column because it will be referred to in the next one. In the meantime, if you see any Fuzzy Logic controllers that are able to hold a boiler's pressure constant, even when you begin to draw off a lot of steam, or a truck speed limiter that can go up hills without slowing down, please let me know where.

All for now. / Comments invited!
RAP / Robert A. Pease / Engineer

rap@galaxy.nsc.com—or:

Address:
Mail Stop D2597A
National Semiconductor
P.O. Box 58090
Santa Clara, CA 95052-8090

References:

1. Pease, R.A., *Troubleshooting Analog Circuits*, 1991, p. 126-127.
2. St. Clair, David, *Controller Tuning and Control Loop Performance*, Straight-Line Controls, 3 Bridle Brook Lane, Newark, DE 19711-2003 (About $16). The URL is *http://members.aol.com/pidcontrol/booklet.html*.
3. McNeill and Freiberger, *Fuzzy Logic*, 1994, p. 107-116.

P.S. Would it be easy to make a similar controller for a steam boiler? Not QUITE that easy, because there may be a LARGE time lag between turning up the heat and seeing an increase in pressure. If there were some data on this lag, it would be fairly easy to design a simple PID with anti-wind-up, which could easily outperform a Fuzzy Logic controller in any specific application—*unless* the Fuzzy Logic controller had full-featured P, D, and I terms in its control. Still, controlling a boiler tends to be more difficult than servoing around an integrator, because the thermal lags can cause the loop gain to roll off MORE STEEPLY than 6 dB/octave.

Of course, you would want to design that controller so that no overshoot can cause the maximum steam pressure to be exceeded, whether the boiler is nearly *empty* or nearly *full* of hot water. Yes, the safety valve can release excess pressure, but it's nice to avoid that, as well as to avoid wasting energy.

The speed limiter for trucks won't have large, unspecified lags of many seconds, as any well-designed electrical, hydraulic, or pneumatic actuator can cut back on the throttle in a small part of a second. Even if the speed limiter is slow or nonlinear, the PID controller is nimble enough. The *lag* of the actuator is compensated for by the *lead* of the PID controller. If the gain of the actuator isn't linear, then the PID controller will have enough gain margin to accommodate that too.

BOB PEASE *obtained a BSEE from MIT in 1961 and is Staff Scientist at National Semiconductor Corp., Santa Clara, Calif.*

What's All This Fuzzy Logic Stuff, Anyhow? (Part V)

Okay, I showed you a Paper Study in the last column (ELECTRONIC DESIGN, *Nov. 6, p. 146*). It was a schematic diagram of a servo controller that I hadn't yet built. I asked you to save that column because you'll need it this month. I told you it was pretty likely to work, and work well. Did it? Yes, it worked very well. Almost perfectly.

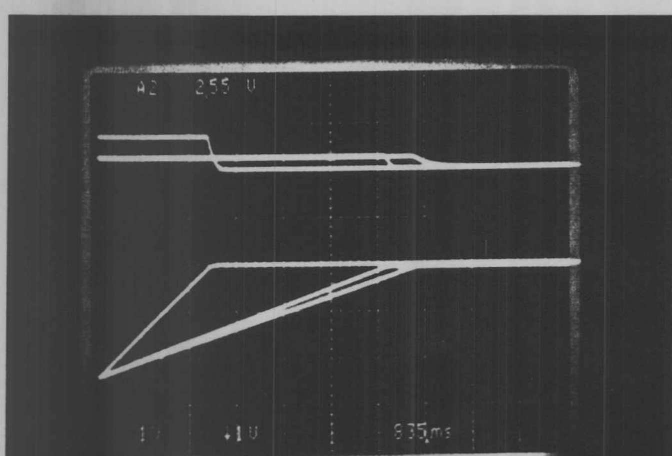
First, I built the "model" of the truck *(Fig. 2 of the previous column)*, and by turning the pots, I could make the "truck's speed" increase (by "stepping on the gas") and decrease, as expected. Hey, a truck trying to pick up speed is JUST like an integrator. (Well, like a *leaky* integrator, because the faster it goes, the less it's able to accelerate fast. And if you take your foot off the gas, it tends to slow down.)

Then I built the main PID servo *(Fig. 1 of the previous column)*. With the integrator disconnected, I connected it up to the "truck model" and started it up. The P and D terms alone made a surprisingly crisp servo. All I had to do to get fast settling and no overshoot was to set those two pots at 100%. Swoop, the output "accelerated" right up to "53.4 mph," ±0.5 mph, and held there with no overshoot *(Scope photo 1)*. The vertical scale factor is 2 mph per division, at ≈0.835 seconds per horizontal division. Looking at the three lower waveforms, the one on the left shows the "speed" of a lightly loaded ("8-ton") truck with fast acceleration, and the middle one reveals a heavily loaded ("40-ton") truck, accelerating at a slower rate. The right waveform shows the "8-ton" truck on an "upgrade." You can see that the limiter is stable for all kinds of loads and biases. The upper waveforms are just the outputs of the differentiator. In a short while we'll discuss why it worked so well, and why I picked the R and C values that I did.

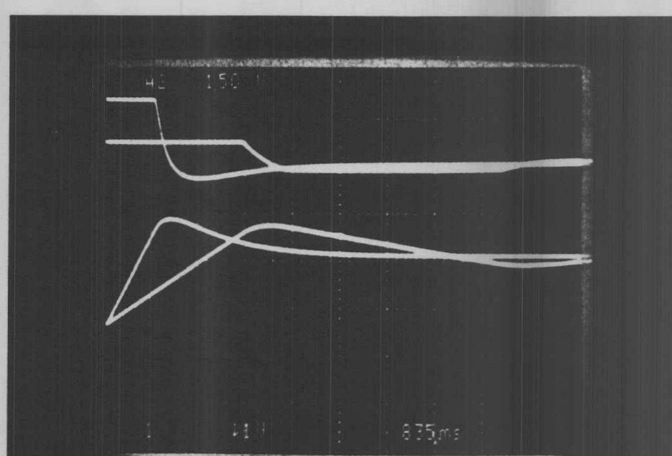
Now, with just the P and D amplifiers connected, the gain was not infinite. The proportional path had a gain of about 5 × 16 = 80. Therefore, if one was "driving uphill" as fast as it would go on a steep upgrade, there would be a little error, perhaps –0.3 mph (i.e., 53.1 mph), which was necessary to let the throttle go nearly wide open (–7 V). Conversely, if the "truck" was on a "downgrade," the speed error might be +0.3 mph, because the main servo path had to pull the gas pedal "up" until it was nearly off (–0.2 V, e.g.)—even if the driver kept his foot FLOORED. Some people might say that ±0.3 mph with no overshoot is pretty good. It's acceptable to many truck drivers. But I wanted to make it much better. (Have you ever seen any Fuzzy Logic (FL) controllers that had accuracy better than that on upgrades and downgrades?)

Next, I connected up the integrator stage. Sure enough, the integrator would "wind up" and cause a 1.8-mph overshoot as the speed limiter was starting to cut in—up to 55.2 mph—exactly as I had expected. See Scope photo 2 with the same two "trucks." But after it settled, the integrator would trim the error to *exactly* 53.4 mph, as desired. Not too bad!

Then I added the anti-wind-up circuit. Son of a gun, that worked, too! It kept the integrator's output at 0 V until the speed limiter began to cut in. Now, the speed would rise right up to 53 mph ±0.2 mph, and promptly settle out to 53.4 mph, exactly, within better than 0.1 mph. That's what I expected



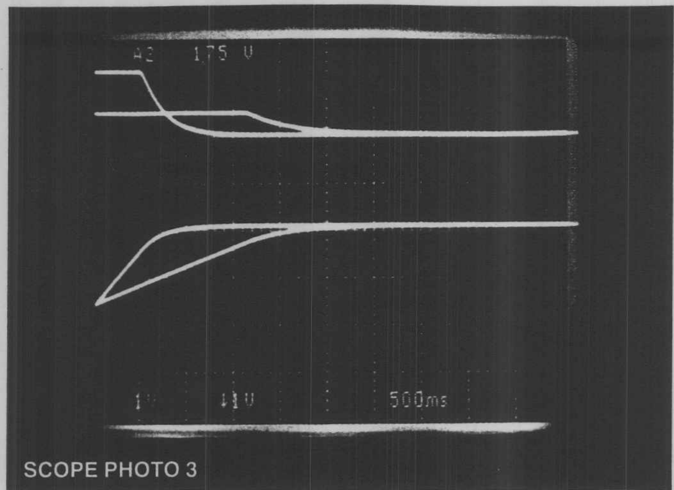
SCOPE PHOTO 1



SCOPE PHOTO 2

(Scope photo 3).

I fooled around with higher gain and lower gain, and things worked as expected. I fooled around with *heavier* versus *lighter* trucks. I even got data (and a scope photo) on an "88-ton truck," but we decided not to print it because it was very well-behaved and *boring*. The control was always well-behaved. I tried cutting back on the gain for the D term and got some slow overshoot. It wasn't really BAD, maybe 0.5 mph. That's what I had anticipated. Then I set that gain back to 100%.



SCOPE PHOTO 3

I varied all of the gains for the integrators AND everything else. The response was pretty clean in every case. I didn't have to make any major or minor adjustments to get the response to be well-behaved. Even when I changed the truck's gear ratio, it was well-behaved. You may notice, though, that the settling of the limiter wasn't quite as "crisp" when the integrator was added, compared to Scope photo 1. There's a little "settling tail." Maybe the PID system isn't perfect, but it's pretty good, and it settles well inside 0.1 mph error in just a couple of seconds.

What components did I change in Figure 1 of the previous column? All three pots wanted to be up near 100%. The input resistance for the integrator worked slightly better at 200 kΩ than at 1 MΩ. The input resistor for the P term still worked well if I paralleled it with another 200 kΩ. That means I had a lot of safety margin, but I left it at 200 kΩ. I didn't have to change the basic differentiator much. But the resistor from the D term to the output wanted to be changed from 27 kΩ to 56 kΩ, so that the differentiator couldn't overpower the other terms and turn off the anti-wind-up too early. In the anti-wind-up circuit, the 1-µF capacitor in the detector that fed the limiter worked slightly better when changed to 0.1 µF *(Fig. 4 of the previous column)*.

I set up some more experiments with and without the anti-wind-up circuit. If the anti-wind-up circuit wasn't provided, it sure did overshoot significantly—not bad enough to fail a test, but bad enough to look crummy.

Some proponents of FL have stated flatly that it would be impossible for any conventional controller, such as a PID type, to cover such a wide range without oscillating. But you just witnessed how it worked well in two worst-case conditions: heavily loaded, and lightly loaded. It also worked at in-between values.

Some FL experts claimed that a conventional or PID approach couldn't possibly work well if the controller wasn't linear. I built some nonlinear circuits in the controller, so the gain at the controller input could be halved or doubled at low throttle settings or at high. It was impossible to see any difference in the results.

In fact, I know one guy who started out his planning for an FL system by building an analog system, which is easy to get a feel for.

Now, that circuit I showed in the previous column was designed to be very versatile—and it was. But it was also quite complicated. So, my plan was to determine the correct coefficients of gain in *that* circuit, and then execute that function with a much simpler circuit *(New Fig. 1)*. This controller is to be hooked up to the basic Figure 2 of the previous column. I engineered this to work the same as the old Figure 1, but with simpler capabilities and less expensive parts.

Note that this circuit is definitely simpler. Instead of using four op amps to make the null amplifier and PID controller, this circuit only requires a dual op-amp. Rather than employing four op amps to make a detector and anti-wind-up circuit, I used a couple of transistors and 1/4 CD4066 to reset the integrator and keep it from causing any errors.

Will this circuit work well under all road conditions with all loads? I'm sure it will. Just because I haven't yet built it does NOT mean that it's NOT certain to work.

Next hack: could we use this controller to act as a speed limiter for a high-performance car? How about a vehicle like a Corvette or a hot Mustang that could accelerate at around 0.5 G even at 50 or 80 mph? Perhaps. But obviously, the input to the throttle controller wouldn't work well with 0.3 seconds of lag. It would require more like 0.1 second.

I COULD chop the 1-µF feedback capacitor in the vehicle's model to 0.1 µF to simulate a high power-to-weight ratio. Plus, I could cut the lag of the controller actuator and run this in real time.

But I actually decided that this is an analog computer, and to simulate a car 10 times faster, I would just declare the time scale changed by a factor of 10. I had proven already that it worked, so I didn't have to rebuild the circuit. This was reasonable, especially because nobody wants to buy such a speed limiter.

In conclusion, when an EXPERT on FL claims to have a scheme that's much better than any *conventional controller*, he may not know that he's only comparing his FL system to a mis-applied or far-off-optimum conventional circuit or system. For years,[1,2,3] I have been saying that in most cases, if you find out how the comparison is made, you may be able to show that a conventional (PID or similar) controller or system can actually do better than the best available FL-based system.

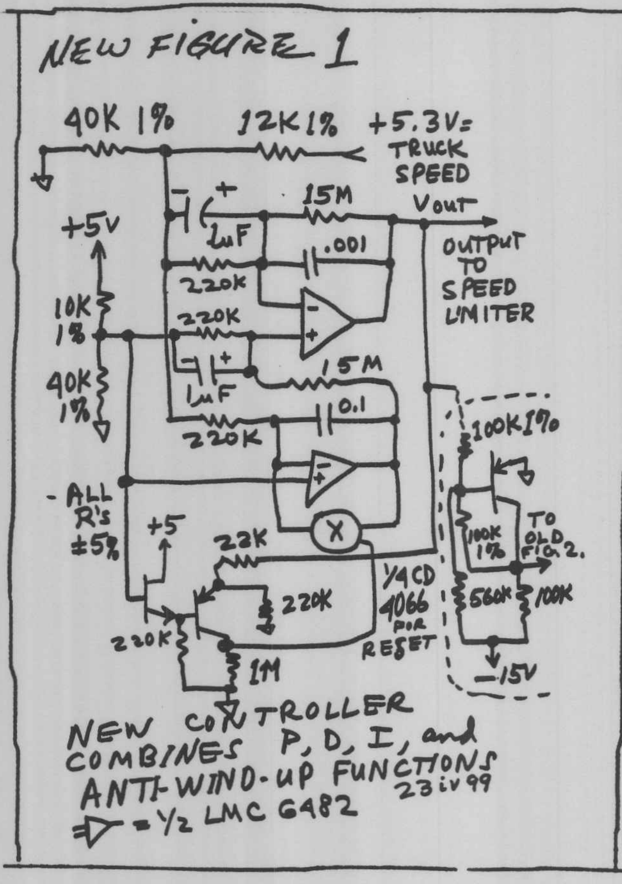
This is partly because it's sometimes not easy to optimize a FL system. It might take a lot of work to get the FL system working well, especially if things are complicated or nonlinear. If an FL controller has P and D terms but

not any I term, then it might have trouble competing with a well-designed PID system because it doesn't have infinite gain at dc.

I also have said previously that I often agree that some systems based on FL are excellent, compared to conventional systems. BUT—not the one in the article that raised all of these questions.

Furthermore, it's true that FL may provide advantages if there's a lot of nonlinearity in the system, or if the controller will work better if it is nonlinear. In that case, the FL system might work quite well—after a lot of study and work. Just don't forget that the conventional system also can have nonlinearities built in.

How well does my PID controller respond when you ask it to respond to an upgrade or downgrade? Does it respond as smoothly and precisely as the FL controller? I would love to plan some comparisons. I have yet to connect my controller to a real vehicle. But at least the model of the truck worked well, with plenty of gain to hold negligible error on "upgrades" and "downgrades."



Plus, I didn't need many trims or adjustment to go from one kind of "truck" to another. My standard controller was well-behaved in each case, even if the "actuator" was nonlinear with steeper gain at some settings and less at others.

Now, do you have to purchase op amps and 1-µF capacitors in order to take advantage of my best controller scheme? Heavens, no. You could execute my PID with anti-wind-up with any number of PID controllers. It could be executed with analog circuits, straight digital, DSP, or even FL. But if you want it to work well, you need to *understand* the signals, the circuits, and the principles behind the analog circuit. If you don't understand all of that, then you might say something foolish, like: "Conventional controllers are OBVIOUSLY inferior. They couldn't POSSIBLY perform this task, so we won't even try one. We'll just build an FL controller." Bad idea!

In fact, I know one guy who started out his planning for an FL system by building an analog system, which is easy to get a feel for. Then he took the best analog system and added refinements, nonlinearity, etc.—and executed it all in FL.

Am I going to put a speed limiter on my Beetle to keep it down below 72 mph? Heck, no. That wouldn't be safe. I have better ideas.

So, what's my point here? As I often say, "The sudden cessation of *my* stupidity is worthy of some note." And if you see FL Experts claiming that they can do things that nobody else can, QUESTION AUTHORITY.

Is it just possible that they have an "excellent system" that's excellent only in comparison to a mis-applied "conventional" controller? Is it possible that they have designed a steam-pressure controller that won't regulate pressure when you begin drawing steam? Is it possible that they have designed a truck-speed controller that will hold the speed constant only until it comes to an upgrade?

Oh yeah, I was going to explain why I picked the components that I did. The "truck" acts as an integrator with unity gain at 3 mHz from the "gas pedal" to the voltage that represents the speed. If I add a gain of 80, it would make the loop have unity gain at 240 mHz. Then there's a lag of 0.3 seconds (estimated) added into the controller path.

That's seen as a lag at a frequency of 0.5 Hz. If I can keep the loop closure below 0.24 Hz, that shouldn't be too close. It might cause a bit of ringing, but I can tweak that out by adding the differentiator.

The lightly loaded truck now looks like an integrator (i.e., just like an op amp) with unity-gain crossover at 0.24 Hz, plus an extra lag at 0.5 Hz. If the main feedback path is a gain of 80, it would be wise to add a lead (gain boost) at frequencies above 0.5 Hz, which means a differentiator with a gain of 80 at 0.5 Hz. That's what you get with a preamp gain of 5, feeding a differentiator at 3.3 seconds. The effective differentiator is at 16.5 seconds, or 60 milliradians/s, or 9 mHz. When will this gain get up to 80? At about 720 mHz. Not bad! That's what I chose from scribbling on the back of envelopes. It worked the first time with no overshoot. This is for the worst case, with a lightly loaded truck. For a truck loaded much heavier, such as five or 10 times heavier, the integrator has a unity-gain crossover at 0.048 or 0.024 Hz. This kind of "op amp" or "integrator" is even easier to close the loop around. It's even more stable.

All for now. / Comments invited!
RAP / Robert A. Pease / Engineer
rap@galaxy.nsc.com—or:

Mail Stop D2597A
National Semiconductor
P.O. Box 58090
Santa Clara, CA 95052-8090

References:

1. Pease, Robert A. "What's All This Fuzzy Logic Stuff, Anyhow?" *Electronic Design*, May 13, 1993, p. 77.

2. Pease, Robert A. "What's All This Fuzzy Logic Stuff, Anyhow? (Part II)," *Electronic Design*, Nov. 1, 1993, p. 95.

3. Pease, Robert A. "What's All This Fuzzy Logic Stuff, Anyhow? (Part III)," *Electronic Design*, Nov. 11, 1993, p. 105.